BEA WebLogic® Integration

Managing WebLogic Integration Solutions

Version 10.2
Document Revised: March 2008

Contents

CHAPTER 1

Managing WebLogic Integration Solutions: Tools and Tasks

This section provides an overview of the tools and tasks involved in managing WebLogic Integration solutions. The following topics are provided:

- WebLogic Integration Management Tools
- WebLogic Managed Beans
- WebLogic Integration Management Task Reference

Note: Throughout this section, the focus is on administrative tasks and tools that are specific to WebLogic Integration. For more information, see *Using the WebLogic Integration Administration Console*.

WebLogic Integration Management Tools

The following tools are available to support WebLogic Integration administration:

- WebLogic Configuration Wizard

  A WebLogic Server® domain is a collection of WebLogic Server resources managed as a single unit. Every domain includes one and only one administration server; any other WebLogic Server instances in the domain are managed servers. The WebLogic Configuration Wizard can be used to assist you in creating and configuring domains to support the development and deployment of WebLogic Integration solutions. See "Creating or Extending Server Domains" on page 1-6 for a quick reference guide to the tasks and related documentation.

- WebLogic Integration Administration Console

The WebLogic Integration Administration Console is a Web application hosted by the administration server in a domain. You access the console from any machine on the local network that can communicate with the WLI administration server through a Web browser. The WebLogic Integration Administration Console allows you to manage and monitor the entities and resources required for your WebLogic Integration applications. For more information about WLI Administration Console, see *Using the WebLogic Integration Administration Console*.

- WebLogic Worklist Console

  The WebLogic Worklist Console is a Web application hosted by the administration server in a domain. You access the console from any machine on the local network that can communicate with the administration server through a Web browser. The console allows administrators to perform WebLogic Server configuration and monitoring tasks without having to learn the JMX API or the underlying management architecture, and manage WebLogic Integration solutions. A list of all the tasks that can be performed from the console is provided in *Using the Worklist Console*.

- WLShell

  This utility provides simplified access to MBeans in WebLogic Server through a scripting language (see the following section, "WebLogic Managed Beans"). It provides a shell-like interface to MBeans in the active WebLogic domain and a Graphic User Interface (GUI) explorer for inspecting MBeans. Using WLShell, you can easily navigate the MBean hierarchy, view configuration and runtime properties, and execute operations such as `get`, `set`, `invoke`, `mkdir`, and `rmdir`. The script support includes loops and conditionals. For more information about this freeware tool, visit http://www.wlshell.com. You can also obtain WLShell from the dev2dev.com.

- Trading Partner Management Bulk Loader

  This utility allows is a command line tool that allows you to import, export, and delete trading partner management (TPM) data. To learn more about this utility, see Chapter 5, "Using the Trading Partner Bulk Loader."

In addition to these tools, WebLogic Server provides a number of tools with which you should be familiar. For more information about the WebLogic Server tools, see *System Administration for BEA WebLogic Server*.

Note: Items or tools on dev2dev.com are listed for your convenience and are not supported by BEA Customer Support.

WebLogic Managed Beans

Resources within a domain use Java Management Extensions (JMX) Managed Beans (MBeans) to expose their management functions.

An MBean is a concrete Java class that is developed per JMX specifications. It can provide getter and setter operations for each management attribute within a managed resource along with additional management operations that the resource makes available. MBeans that expose the configuration data of a managed resource are called *Configuration MBeans*, while MBeans that provide performance metrics and other information about the runtime state of a managed resource are called *Runtime MBeans*.

For more information about WebLogic Server managed resources and MBeans, see Overview of WebLogic JMX Services in *Programming WebLogic Management Services with JMX*.

For more information about the WebLogic Integration MBeans, see the following packages in the WebLogic Integration Javadoc:

- com.bea.wli.management.configuration
- com.bea.wli.management.runtime
- com.bea.wli.tpm.management.configuration
- com.bea.wli.tpm.management.runtime
- com.bea.wli.management.deployment
- com.bea.wli.management.runtime

Programmatically Accessing WebLogic Integration MBeans

The `weblogic.management.MBeanHome` interface is the most convenient way to access the JMX MBean Server that resides on each WebLogic Server in a domain. You can access the Administration `MBeanHome` interface from the JNDI tree of the Administration Server as described in "Using JNDI to Retrieve an MBeanHome Interface" in Accessing WebLogic Server MBeans.

Listing 1-1 shows how you can access the ProcessRuntimeMBean interface:

Listing 1-1 Programmatically Accessing ProcessRuntimeMBean

```
Environment env = new Environment();
env.setSecurityPrincipal("weblogic");
env.setSecurityCredentials("weblogic");
Context ctx = env.getInitialContext();
MBeanHome home = (MBeanHome)ctx.lookup(MBeanHome.ADMIN_JNDI_NAME);
System.out.println("Got the Server-specific MBeanHome: " + home);
Set s = home.getMBeansByType("ProcessRuntime");
Iterator it = s.iterator();

try

{

              while (it.hasNext())

               {

                ProcessRuntimeMBean bean = (ProcessRuntimeMBean)it.next();
                ProcessInstanceQuery query = new
                 ProcessInstanceQuery();query.setServiceURI(context.getSe
     rvice().getURI());
                 ProcessInstanceQueryResult info =
                 bean.getProcessInstances(query);
                 String[] instances = info.getInstanceIds();
                 System.out.println(instances[0]);

                 }
}

         catch (Exception ex)

         {

         System.out.println(ex);

         ex.printStackTrace();
         }
```

WebLogic Integration Management Task Reference

This section provides references to the instructions and background information required to perform the most common WebLogic Integration administrative tasks:

- Creating or Extending Server Domains
- Managing Database Resources
- Deploying Integration Solutions
- Securing WebLogic Integration Resources
- Managing Process Types
- Monitoring Process Instances
- Monitoring Message Broker Channels
- Creating and Managing Event Generators
- Managing WebLogic Integration Tracking and Reporting Data
- Creating Business Calendars and Assigning Them to Users or Groups
- Managing and Monitoring Worklist Task Plan
- Managing Trading Partner Integration
- Managing XML Cache Instances

Majority of these tasks can be performed using the WebLogic Integration Administration Console.

Some of the tasks must be performed using other tools, and in some cases, you must directly edit a configuration file. You can use this section as a roadmap to the task-specific information that can be found in the following resources:

Table 1-1 Resources

Document Title
Managing WebLogic Integration Solutions
Using the WebLogic Integration Administration Console
Deploying WebLogic Integration Solutions

Table 1-1 Resources

Overview of WebLogic Server System Administration
Creating WebLogic Configurations Using the Configuration Wizard
Security in WebLogic Platform 10.0
Introducing Trading Partner Integration
Using the Worklist

Throughout this reference section, it is assumed that the WebLogic Integration Administration Console is to be used as the primary management tool. As described in "WebLogic Integration Management Tools" on page 1-1, alternative utilities, such as the SNMP Agent or WLShell, can be used to perform many tasks.

Creating or Extending Server Domains

A domain includes one or more instances of WebLogic Server and may include WebLogic Server clusters. WebLogic Integration is a collection of applications and resources—EJBs, Web applications, JDBC connection pools, and so on—that are deployed in a domain to provide a unified platform for developing and deploying comprehensive business integration solutions. A first step in the development or deployment of a WebLogic Integration solution is to create a suitable domain.

The following table provides a roadmap to the information you need to create or extend a development or production (running in "noniterativedev" mode) domain.

Table 1-2 Roadmap to Create or Extend a Domain

To. . .	Refer to. . .	The reference provides. . .
Create a basic single server or clustered domain	The following sections of *Creating WebLogic Configurations Using the Configuration Wizard:* • Overview of the WebLogic Configuration Wizard and Configuration Template Builder • Template Reference: Basic WebLogic Server Domain • Template Reference: WebLogic Integration Extension Template • Creating a New WebLogic Domain • Configuring Managed Servers, Clusters, and Machines • Extending Domains • How Do I? . . . Creating XA Domains Using Configuration Templates	General information in the overview about WebLogic Server domains and how to use the Configuration Wizard. The template reference sections provide information about the default WebLogic Integration templates provided by the Wizard. The remaining sections provide procedural information.
Prepare a production domain	The following sections of *Deploying WebLogic Integration Solutions*: • Introduction • Understanding WebLogic Integration Clusters • Configuring a Clustered Deployment Related tasks and references are provided in "Deploying Integration Solutions" on page 1-8 and "Securing WebLogic Integration Resources" on page 1-8.	Describes key domain resources and deployment tasks in the introduction. It also provides a discussion of the roles played by system administrators, deployment specialists, and database administrators. "Understanding WebLogic Integration Clusters" provides background and "Configuring a Clustered Deployment" provides step-by-step procedures.
Create the database tables required by WebLogic Integration	Chapter 2, "Configuring a Production Database." Additional references are provided in the following section, "Managing Database Resources."	Describes the scripts provided to create the tables required by WebLogic Integration.

Managing Database Resources

For general information about managing database resources for WebLogic Platform, see *Managing WebLogic Platform Database Resources*.

For information about creating the tables required by WebLogic Integration, see Chapter 2, "Configuring a Production Database."

Deploying Integration Solutions

For information about deploying an integration application from the environment (running in iterative development mode), see Building and Deploying WebLogic Integration Applications in *Guide to Building Business Process.*

For the background information and procedures required to configure a production environment and deploy integration solutions, see *Deploying WebLogic Integration Solutions*.

Securing WebLogic Integration Resources

This section focuses on security tasks and references that are specific to WebLogic Integration.

Table 1-3 provides a roadmap to the information you need to secure WebLogic Integration resources.

Table 1-3 Roadmap to Secure WebLogic Integration Resources

To. . .	Refer to. . .	The reference provides. . .
Verify security provider requirements	Security Provider Requirements for User Management in the *Using the Worklist Console*.	Requirements.
Manage users, groups, and roles	User Management in the *Using WebLogic Integration Administration Console*	Step-by-step procedures for adding, deleting, or updating users, groups, and roles.
Learn about users, groups, and roles in WebLogic Integration	WebLogic Integration Users, Groups, and Roles section in the *Using WebLogic Integration Administration Console.*	A brief overview.
	Default Groups, Roles, and Security Policies section in the *Using WebLogic Integration Administration Console*.	Description of built in groups, roles, and security policies.

Table 1-3 Roadmap to Secure WebLogic Integration Resources

To. . .	Refer to. . .	The reference provides. . .
Configure the role required to invoke process operations	Process Security Policies section in the *Using The WebLogic Integration Administration Console*.	WebLogic Integration Administration Console procedures.
Configure the roles required to subscribe or publish to message broker channels	Setting Channel Security Policies section in the *Using The WebLogic Integration Administration Console.*	WebLogic Integration Administration Console procedures.
Configure the role authorized to create worklist tasks	User Management section in the *Worklist Console Help.*	Using Worklist Console procedures.
Manage the password store	The following sub-sections of the System Configuration section in *Using WebLogic Integration Administration Console*: • Password Aliases and the Password Store • Adding Passwords to the Password Store • Listing and Locating Password Aliases • Changing the Password for a Password Alias • Deleting Passwords from the Password Store	WebLogic Integration Administration Console procedures.
Securing resources for trading partner integration	The following sub-sections of Trading Partner Management section in *Using the WebLogic Integration Administration Console*: • Trading Partner Integration Security • Example: ebXML Security Configuration • Example: RosettaNet Security Configuration	Trading partner security

Managing Process Types

Process types can be monitored from the WebLogic Integration Administration Console. For a description of the Process Configuration module, and step-by-step procedures for the various management tasks, see Process Configuration in *Using the WebLogic Integration Administration Console*.

You can also access the graphical view of a process type from other HTTP clients. See Chapter 4, "Accessing Process Graphs from HTTP Clients."

Monitoring Process Instances

Process instances are monitored from the WebLogic Integration Administration Console. For a description of the Process Instance Monitoring module, and step-by-step procedures for the various monitoring tasks, see Process Instance Monitoring in *Using the WebLogic Integration Administration Console*.

You can also access the graphical view of a process instance from other HTTP clients. See Chapter 4, "Accessing Process Graphs from HTTP Clients."

Monitoring Message Broker Channels

Message broker channels are monitored from the WebLogic Integration Administration Console. For more information about Message Broker module, and step-by-step procedures for the monitoring tasks, see Message Broker in *Using the WebLogic Integration Administration Console*.

Creating and Managing Event Generators

WebLogic Integration provides native event generators, including JMS, Email, File, and Timer event generators. These event generators are typically used to start a business process based on events, such as the receipt of email or a new file appearing in a directory. WebLogic Integration Table 1-4 provides a roadmap to the information you need to manage event generators.

Table 1-4 Managing Event Generators

To. . .	Refer to. . .	The reference provides. . .
Get information about the JMS, Email, File, Timer, MQ Series, RDBMS, and HTTP event generators.	The Event Generators section in *Using The WebLogic Integration Administration Console*	Introduction to the event generators (which publish messages to Message Broker channels in response to system events).
	"Message Broker Resources" and "Event Generator Resources" in Introduction in *Deploying WebLogic Integration Solutions*.	

Table 1-4 Managing Event Generators

To. . .	Refer to. . .	The reference provides. . .
Create and deploy a File, Email, JMS, Timer, MQ Series, RDBMS, or HTTP event generator	Creating and Deploying Event Generators section in *Using The WebLogic Integration Administration Console*	WebLogic Integration Administration Console procedures.
	"Deploying Event Generators" in Understanding WebLogic Integration Clusters in *Deploying WebLogic Integration Solutions*.	Information about event generator targeting and error handling.
Manage the JMS, Email, File, Timer, MQ Series, RDBMS, or HTTP event generators	Event Generators section in *Using The WebLogic Integration Administration Console*	Procedures for updating channel rules, or deleting suspending, or resuming an event generator.
Configure JMS event generators to consume the first element under the `<SOAP:Body>` element.	The description of the `wli.jmseg.EatSoapActionElement` element in wli-config.properties Configuration File in *Deploying WebLogic Integration Solutions.*	Configuration property description.

Managing WebLogic Integration Tracking and Reporting Data

The following table provides a roadmap to the information you need to manage WebLogic Integration tracking and reporting data.

Table 1-5 Managing Tracking and Reporting Data

To. . .	Refer to. . .	The reference provides. . .
Learn about the tracking data	• Process Tracking Data • Reporting and Purging Policies for Tracking Data • Managing Process Tracking Data section in *Using The WebLogic Integration Administration Console*	Descriptions of the tracking data available, the tracking levels that can be set, and the related management tasks, such as configuring a reporting database for offline storage or defining the schedule for purging the data from the runtime database.
Query the reporting data tables	• Chapter 3, "Querying WebLogic Integration Reporting Data."	Descriptions of key tables and example queries.
Set the system-level policies for purging tracking data from the runtime database.	• Viewing the Configuration for Tracking, Reporting, and Purging Data • Configuring the Reporting Data and Purge Process • Configuring the Default Tracking Level and Reporting Data Policy sections in *Using The WebLogic Integration Administration Console*	WebLogic Integration Administration Console procedures.
Configure the Reporting Data Datastore	Configuring the Reporting Datastore in *Using The WebLogic Integration Administration Console*	WebLogic Integration Administration Console procedure.
Configure the tracking level for a process	Viewing and Changing Process Details in *Using The WebLogic Integration Administration Console*	WebLogic Integration Administration Console procedure.
Configure the tracking level for business messages	Configuring the Mode and Message Tracking section in *Using The WebLogic Integration Administration Console*	WebLogic Integration Administration Console procedure.

Creating Business Calendars and Assigning Them to Users or Groups

Most of the management tasks associated with business calendars are completed from the WorkList Administration Console. For a description of the Business Calendar Configuration

module, and step-by-step procedures for the various management tasks, see Business Calendar Configuration section in *Using Worklist Console.*

Managing and Monitoring Worklist Task Plan

Most of the management tasks associated with the worklist can be completed from the WebLogic Integration Administration Console. For a description of the Worklist Administration module, and step-by-step procedures for the various management tasks, see Worklist Administration section in *Using Worklist Console*.

Custom worklist interfaces can also provide administrative and management functionality. For more information about custom worklist interfaces, see Using and Customizing User Portal in *Using Worklist.*

For more information about worklist operations, see the following sections of *Using Worklist Console*.

- Introduction
- Creating and Managing Worklist Tasks

Managing Trading Partner Integration

Most of the trading partner integration management tasks are completed from the WebLogic Integration Administration Console. For more information about the Trading Partner Management module, and step-by-step procedures for the various management tasks, see Trading Partner Management in *Using WebLogic Integration Administration Console*.

You can also use the Bulk Loader command line utility to import and export trading partner management data. See Chapter 5, "Using the Trading Partner Bulk Loader."

For more information about securing trading partner integration applications, see Securing WebLogic Integration Resources .

Managing XML Cache Instances

The XML Cache stores XML metadata documents. When you are designing a business process, XML Cache Controls are used to retrieve the XML documents stored in the XML Cache. You use the XML Cache module to create and maintain the XML metadata documents stored in the XML Cache. For more information about XML Cache module, and step-by-step procedures for the various management tasks, see XML Cache section in *Using The WebLogic Integration Administration Console*.

CHAPTER 2

Configuring a Production Database

Configure the database to include the tables required by WebLogic Integration, before you start preparing a production environment for WebLogic Integration. To allow your database administrator to manage the process, the tables required are not created automatically. This section provides information about the scripts available to create the tables.

The SQL scripts that create the database tables used by WebLogic Integration can be found in the following directory:

```
BEA_HOME\wli_10.2\dbscripts\vendor\
```

In this path, *BEA_HOME* represents the WebLogic Platform home directory, and *vendor* represents the vendor of the database you will be using in production mode. Table 2-1 describes the scripts.

Table 2-1 SQL Scripts and Descriptions

Script filename	Description
`wli_runtime.sql`	SQL that creates tables involved in WebLogic Integration runtime activity.
`wli_runtime_drop.sql`	SQL that drops tables created by `wli_runtime.sql`. **Note:** All runtime data is destroyed.
`wli_archive.sql`	SQL that creates tables used to store WebLogic Integration data for reporting and analysis.

Table 2-1 SQL Scripts and Descriptions

Script filename	Description
`wli_archive_drop.sql`	SQL that drops tables created by `wli_archive.sql`. **Note:** All runtime data is destroyed.
`upgrade_wli_runtime_sp4_9.2.sql`	SQL that runs `upgrade_wli_runtime_sp4_9.2.sql` in order to have an additional wli table WLI_EVENTGEN_STATE in upgraded WLI 10.2 Domain.
`worklist_runtime.sql`	SQL that creates tables involved in Worklist runtime activity.
`upgrade_worklist_runtime_8.5_9.0.sql`	SQL that runs `upgrade_worklist_run`
`worklist_runtime_drop.sql`	SQL that drops tables created by `worklist_runtime.sql`.
`worklist_reporting.sql`	SQL that creates tables to store reports created by worklist.
`worklist_reporting_drop.sql`	SQL that drops report tables created by worklist.

Use your preferred SQL tool to run the scripts to create or drop the WebLogic Integration tables in your production database.

Note: In addition to the WebLogic Integration tables, you must also create the database tables that store conversational state information. For more information about database tables, see Preparing a Database.

If you have trading partner management data, you can use the Bulk Loader to import the information. Refer to Chapter 5, "Using the Trading Partner Bulk Loader."

CHAPTER 3

Querying WebLogic Integration Reporting Data

As described in the About System Administration section in *Using The WebLogic Integration Administration Console*, the reporting database tables contain information regarding events that occur during the execution of processes. These tables are created by the SQL commands in the file `wli_archive.sql` described in Chapter 2, "Configuring a Production Database.".

To generate reports from reporting database, you will need to run SQL queries. This section describes useful tables and provides example queries:

- The WLI_PROCESS_EVENT_ARCH Table
- The WLI_DOCUMENT_DATA Table
- Example Queries

The WLI_PROCESS_EVENT_ARCH Table

As a process executes, events are generated that track its execution. The events generated depend on the tracking level configured (see Managing Process Tracking Data section in *Using WebLogic Integration Administration Console*). For example, if the tracking level for a process is set to **Full** or **Node**, two events, start node and end (or abort) node, are generated by each node.

If the process tracking data is transmitted to the reporting database, each event is stored as row in `WLI_PROCESS_EVENT_ARCH` table. The row contains the process name (a URI value), process instance ID, process event type (see com.bea.wli.management.archiving.TrackingEventType), and other values.

The PROCESS_LABEL column is set only for events generated by calls to:

```
JpdContext.setProcessLabel(String)
```

The WLI_DOCUMENT_DATA Table

Invoking the JpdContext.trackData(*payload*) method generates an event of type EVENT_TYPE_PROCESS_LOG. If the data is transmitted to the reporting database, each event is stored as a new row in the WLI_DOCUMENT_DATA table. The *payload* is stored in the DATA column of that table, and the EVENT_DATA_ID column provides a link to the event in the WLI_PROCESS_EVENT_ARCH table.

In addition to containing the results of trackData(), the WLI_DOCUMENT_DATA table contains unhandled exceptions generated by the process instance and business message payloads (if business messages are tracked).

The valid types for WLI_DOCUMENT_DATA.TYPE are defined in com.bea.wli.management.archiving.DocumentDataType.

For additional information about:

- The trackData() method, see JpdContext interface.
- Configuring business message tracking, see Configuring the Mode and Message Tracking section in *Using The WebLogic Integration Administration Console*.

Example Queries

The following example queries are provided:

- Get the Average Elapsed Time for a Process
- Get the Average Elapsed Time for a Node
- Get Results of the trackData() API

Note: See com.bea.wli.management.archiving.TrackingEventType for the constant field value for each event type. For example, in the following examples, 3 corresponds to EVENT_TYPE_PROCESS_ACTIVITY_END and 20 corresponds to EVENT_TYPE_PROCESS_LOG.

Get the Average Elapsed Time for a Process

To get the average elapsed time for a given process on a given day, the SQL query is:

```
SELECT AVG(EVENT_ELAPSED_TIME) FROM WLI_PROCESS_EVENT_ARCH
WHERE PROCESS_TYPE = PROC_TYPE
AND ACTIVITY_ID = 0
AND EVENT_TYPE = 3
AND (EVENT_TIME >= START_TIME AND EVENT_TIME < END_TIME)
AND DEPLOYMENT_ID IN
       (SELECT MAX(DEPLOYMENT_ID)
       FROM WLI_PROCESS_EVENT_ARCH
       WHERE PROCESS_TYPE = PROC_TYPE)
```

In this query, *PROC_TYPE* should be replaced by a value from the `WLI_PROCESS_EVENT_ARCH` table, and *START_TIME* and *END_TIME* should be literal timestamps.

Get the Average Elapsed Time for a Node

To get the average elapsed time for a given node in a given process on a given day, the SQL query is:

```
SELECT AVG(WPEA.EVENT_ELAPSED_TIME)
FROM WLI_PROCESS_EVENT_ARCH WPEA, WLI_PROCESS_DEF_ARCH WPDA
WHERE WPEA.PROCESS_TYPE = PROC_TYPE
AND WPEA.EVENT_TYPE = 3
AND (WPEA.EVENT_TIME >= START_TIME and WPEA.EVENT_TIME < END_TIME)
AND WPEA.PROCESS_TYPE = WPDA.PROCESS_TYPE
AND WPEA.ACTIVITY_ID = WPDA.ACTIVITY_ID
AND WPEA.DEPLOYMENT_ID = WPDA.DEPLOYMENT_ID
AND WPDA.USER_NODE_NAME = NODE_NAME
AND WPDA.DEPLOYMENT_ID IN
       (SELECT MAX(DEPLOYMENT_ID) FROM WLI_PROCESS_DEF_ARCH
       WHERE PROCESS_TYPE = PROC_TYPE)
```

In this query, *PROC_TYPE* and *NODE_NAME* should be replaced by values from the `WLI_PROCESS_EVENT_ARCH` table, and *START_TIME* and *END_TIME* should be literal timestamps.

Get Results of the trackData() API

To get the result of all `trackData()` calls for a given process type, the SQL query is:

```
SELECT WDD.DATA, WDD.TYPE, WPEA.PROCESS_INSTANCE
FROM WLI_DOCUMENT_DATA WDD, WLI_PROCESS_EVENT_ARCH WPEA
WHERE WDD.EVENT_DATA_ID = WPEA.EVENT_DATA_ID
AND WPEA.PROCESS_TYPE = PROC_TYPE
AND WPEA.EVENT_TYPE = 20
```

In this query, *PROC_TYPE* should be replaced by a value from the `WLI_PROCESS_EVENT_ARCH` table.

CHAPTER 4

Accessing Process Graphs from HTTP Clients

The interactive process graph, and the associated process type or process instance data, which can be viewed from within the WebLogic Integration Administration Console, can also be accessed from other HTTP clients. For more information about process graphs, see Viewing an Interactive or Printable Process Type Graph in *Using WebLogic Integration Administration Console.* This section describes the how to access the process graph. The following topics are provided:

- Supported Clients
- The HTTP Request URL

Supported Clients

The following types of clients are supported:

- Web browsers with an SVG plug-in
- Java client applications using custom SVG tools, such as the Apache Batik toolkit.

The HTTP Request URL

The WebLogic Integration Administration Console Web application accepts HTTP requests (containing Service URI and Instance ID) from a client and returns an SVG document. The client fetches related JavaScript and image files via subsequent requests to the Web application. Both web browser clients (using an SVG plug-in) and Java client applications (using SVG tools) are supported.

The primary command has the following form:

http://localhost:7001/wliconsole/procgraph?com=procgraph&wfuri=ServiceURI&wfid=InstanceID

For example:

`http://localhost:7001/wliconsole/procgraph?com=procgraph&wfuri=`**`%2Fwlitest%2Fmy_process.jpd`**`&wfid=`**`1063226907001`**

If you omit `&instanceid=`*`InstanceID`*, the SVG document for the process type is returned.

The SVG document that is initially retrieved from the Web application references additional resources on the server such as images and JavaScript files. These additional resources are retrieved automatically by most browser plug-ins by processing the `xlink:href` attributes in the SVG document.

CHAPTER 5

Using the Trading Partner Bulk Loader

The Bulk Loader is a command line tool that you can use to import, export, and delete trading partner management (TPM) data. This data includes trading partner profiles, certificates from keystores, service definitions, and service profiles. The Bulk Loader imports an XML representation of TPM data and it exports to an XML file. Validation of the XML input documents is performed using the XSD schemas. The Bulk Loader uses an XML configuration file (`blconfig.xml`) to obtain parameters for connecting to the database and certificate keystores. If the Bulk Loader detects any errors during this procedure, it creates an error log.

The following sections provide information on using the Bulk Loader:

About Using the Bulk Loader

The Bulk Loader command line tool should only be used when the WebLogic Integration server is *not* running. If the WebLogic Integration server is running, all configuration changes to TPM data in the database should be performed through the WebLogic Integration Administration

Console. The WebLogic Integration Administration Console also supports import, export, and bulk delete operations. Using the WebLogic Integration Administration Console for these operations ensures that the running servers in a WebLogic Integration domain have consistent TPM data in their internal TPM memory cache.

For more information, see the following sections in *Using the WebLogic Integration Administration Console*.

- Importing Management Data
- Exporting Management Data
- Deleting Trading Partner Profiles and Services Using Bulk Delete

Schemas

When importing and exporting repository data and trading partner configuration, two XSD schemas are used by the Bulk Loader to validate the imported or exported XML documents. The `TPM.xsd`, which specifies the trading partner information and the `BulkLoaderConfig.xsd`, which specifies database and keystore information and the transaction processing options. These schemas are based on the 2001 XML Schema Definition (XSD).

Both the `TPM.xsd` and `BulkLoaderConfig.xsd` schemas are in the `schema/src` directory inside the `jpd.jar` file. These files are located in the following directory:

```
BEA_HOME\wli_10.2\lib
```

BEA_HOME represents the WebLogic Platform home directory.

To learn about the entities and elements that comprise trading partner management data in the `TPM.xsd` file, see Chapter 6, "Trading Partner Management Schema."

To learn about setting up keystore information and the transaction processing options in the `BulkLoaderConfig.xsd`, see "Transaction Processing Options" on page 5-5 and "Importing or Exporting Certificate Elements" on page 5-6.

Configuring the Bulk Loader Configuration File

The Bulk Loader uses a configuration file (`blconfig.xml`) to get parameters for connecting to the database and certificate keystores. Before using the Bulk Loader, you must modify this file to match your database installation.

The `blconfig.xml` configuration file is located in the following directory:

```
BEA_HOME\wli_102\bin
```

BEA_HOME represents the WebLogic Platform home directory.

Listing 5-1 blconfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkloader-config
  xmlns="http://www.bea.com/2003/03/wli/tpm/bulkloader"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bea.com/2003/03/wli/tpm/bulkloader
BulkLoaderConfig.xsd">
  <database-info>
    <!-- Modify the following to match your database installation
-->
    <url>jdbc:pointbase://localhost:9093/workshop</url>
    <driver>com.pointbase.jdbc.jdbcUniversalDriver</driver>
    <userid>weblogic</userid>
    <password>weblogic</password>
  </database-info>
  <encoding>UTF-8</encoding>
</bulkloader-config>
```

Using the Bulk Loader Command Line Options

The Bulk Loader is located in the following directory:

```
BEA_HOME\wli_102\bin
```

In the preceding line *BEA_HOME* represents the WebLogic Platform home directory.

The Bulk Loader usage is as follows:

```
bulkloader [-verbose] [-config <blconfig.xml>] [-wlibc]
-import <data.xml>
-export <data.xml> [-nokeyinfo] [-select <selector.xml>]
-delete <selector.xml>
```

Table 5-1 summarizes the options for the Bulk Loader commands.

Table 5-1 Bulk Loader Commands

Option	Description
`[-verbose]`	Optional. Use verbose mode to help you troubleshoot problems in your import, export, or delete process.
`[-config <blconfig.xml>]`	Optional. Use to designate an explicit configuration file. Default is `blconfig.xml`. If not using the default, specify the full path of the configuration file.
`-import <data.xml>`	Use to import data. Specify the full path of the TPM file you want to import. To learn more about importing, see "Importing and Exporting Trading Partner Management Data" on page 5-5.

`-export <data.xml>`
`[-nokeyinfo]`
`[-select <selector.xml>]`
	Use to export data. Specify the full path of the TPM file you want to export.
	The `[-nokeyinfo]` option suppresses export of `KeyInfo` elements for trading partner certificates.
	The `-select` option specifies the selector file and `selector.xml` specifies the type of data to be exported. You can use the `selector.xml` file to export all or just selected Trading Partners. This file can also designate that all or selected Services for export. This file must conform to the `TPM.xsd` schema.
	To learn more about exporting, see "Importing and Exporting Trading Partner Management Data" on page 5-5.

`-delete <selector.xml>`	Use to delete data. Specify the full path of the TPM file used for selecting the elements to be deleted. Use `selector.xml` to specify the elements that you want to delete. This file must conforms to the `TPM.xsd` schema. To learn more about deleting, see "Deleting Management Data" on page 5-9.

Importing and Exporting Trading Partner Management Data

You can import or export trading partner management information including certificate data using the Bulk Loader. The Bulk Loader imports an XML representation of the TPM data and it exports an XML file. Before importing or exporting certificates you need to modify the `blconfig.xml` file as described in "Importing or Exporting Certificate Elements" on page 5-6. How to import and export trading partner information is described in the following topics:

Transaction Processing Options

In case of errors or when working with large repositories, you can use two attributes contained in the `BulkLoaderConfig.xsd` schema to control transaction processing. These attributes are `transaction-level="all"` and `transaction-level="default"`. They are under the `<bulkloader-config>` root element. These options provide the same functionality available in the WebLogic Integration Administration Console.

The attribute `transaction-level="all"` performs the following:

- Imports the data in a single transaction. If invalid data is detected the entire transaction is rolled back.
- Exports all trading partner management entities.
- Deletes the data in a single transaction. If invalid data is detected the entire transaction is rolled back.

The attribute `transaction-level="default"` performs the following:

- Imports data using multiple transactions. The import initiates a transaction for each trading partner or service. If invalid data is detected during a transaction for any entity, the import is rolled back for the current transaction only; importing stops with the rolled back transaction.
- Exports the data specified in the `selector.xml` file. (This file must conforms to the `TPM.xml` schema.)

- Deletes the data using multiple transactions. A delete transaction is initiated for each trading partner or service. If an error is encountered during the transaction for any entity, the transaction is rolled back; deleting stops with the rolled back transaction.

General Procedure for Importing and Exporting

This section contains information about importing and exporting trading partner management data.

To import or export trading partner management data:

Before importing or exporting a TPM file, ensure that the TPM file conforms to the `TPM.xsd` schema.

1. On a Windows system, open a command window.
2. In both Windows and UNIX, go to the following directory:

   `BEA_HOME\wli_102\bin`

   In the preceding line, *`BEA_HOME`* represents the WebLogic Platform home directory.
3. Execute the import or export by entering the appropriate commands:

```
bulkloader [-verbose] [-config <blconfig.xml>] [-wlibc]
-import <data.xml>
-export <data.xml> [-nokeyinfo] [-select <selector.xml>]
```

The following shows an example of importing a trading partner XML file:

```
bulkloader -wlibc -import
d:\tradingpartners\profiles\WorldWideTrading.xml
```

This example shows exporting services offered by a remote trading partner:

```
bulkloader -config myconfig.xml -export
exports\NationalTradingServices.xml -select
selectors\NationalTradingSelector.xml
```

Importing or Exporting Certificate Elements

Note: Only the certificates for remote Trading Partners can be imported; certificates for local Trading Partners cannot be imported.

Importing and exporting of certificates, as with other trading partner profile information, is done in XML format. The XML representation of the certificates conforms to the certificate

representation format specified in the W3C XML-Signature Syntax and Processing recommendation, which is available at the following URL:

http://www.w3.org/TR/xmldsig-core/#sec-KeyInfo

The Bulk Loader only supports import or export of certificate data and public keys. The Private Key of certificates is not imported or exported; an administrator must manually perform the transfer of the Private Key. The keystore related information is read from the Bulk Loader configuration file (`blconfig.xml`).

Note: To learn more about the WebLogic Server Keystore, see "WebLogic Keystore Provider-->General" in the Administration Console Online Help.

When the input XML file has certificate elements for a trading partner with `<ds:KeyInfo>` sub-elements, the specified `certificate-key` data is added to the appropriate keystore as designated by the Bulk Loader configuration file.

The Bulk Loader configuration schema (`BulkLoaderConfig.xsd`) includes keystore configuration information. This is an optional element in the schema. The following extract is from the schema definition for the `keystore-info` element:

```
<xs:element name="keystore-info">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="path" type="xs:string"/>
    <xs:element name="password" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="encoding" type="xs:string"/>
```

Passwords for the database and keystore can be initially entered in the `blconfig.xml` file in clear text. After the operation successfully completes, the Bulk Loader encrypts the passwords and re-writes the `blconfig.xml` file with the encrypted form of the passwords.

The `path` element is the absolute file path to the Java KeyStore. The password element is the keystore password.

The following is an example of the Bulk Loader configuration file that includes keystore information.

Listing 5-2 blconfig.xml with Keystore Information

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkloader-config
```

```
    xmlns="http://www.bea.com/2003/03/wli/tpm/bulkloader"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.bea.com/2003/03/wli/tpm/bulkloader
    BulkLoaderConfig.xsd">
  <database-info>
    <url>jdbc:pointbase://localhost:9094/WLIDB</url>
    <driver>com.pointbase.jdbc.jdbcUniversalDriver</driver>
    <userid>PBPUBLIC</userid>
    <password>PBPUBLIC</password>
  </database-info>
  <keystore-info>
    <path>D:\test\peer1KeyStore.pks</path>
    <password>peer1<password>
  </keystore-info>
</bulkloader-config>
```

The following is an example of trading partner information with a client certificate in import-export format.

Listing 5-3 Trading Partner with Client Certificate

```
<trading-partner
  name="ebxml-sender"
  type="REMOTE"
  status="ENABLED">
  <client-certificate name="peer1-en">

<KeyInfo>
  <KeyName>1.2.840.113549.1.9.1=#160d706565723140626265612e636f6d,
    CN=localhost.peer1-en.crt,OU=ECI Division,O=BEA Systems,
    ST=California,C=US
  </KeyName>
  <KeyValue>
    <RSAKeyValue>
      <Modulus>t/kDK6Jezk2e31k2nMQMagPuXsC56df18YW0KRqQa89Q7o/
        H8O8m6LdOH5H0GyYEUBD+jN08lgZqCQMDAZCG6w==</Modulus>
      <Exponent>AQAB</Exponent>
    </RSAKeyValue>
  </KeyValue>
  <X509Data>
  <X509SubjectName>1.2.840.113549.1.9.1=#160d706565723140626265612e636
f6d,
    CN=localhost.peer1-en.crt,OU=ECI Division,O=BEA Systems,
    ST=California,C=US</X509SubjectName>
```

```
  <X509IssuerSerial>
<X509IssuerName>1.2.840.113549.1.9.1=#1610676172696d656c73406265612e
636f6,
  CN=luke.bea.com,OU=WLC Luke,O=ECI Division\, BEA Systems Inc,
  L=San Jose,ST=California,C=US</X509IssuerName>
    <X509SerialNumber>DQ==</X509SerialNumber>
  </X509IssuerSerial>
<X509Certificate>MIICQzCCAe2gAwIBAgIBDTANBgkqhkiG9w0BAQQFADCBqDELMAk
GA1UEB
hMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMSYwJAYDV
QQKEx1
FQ0kgRGl2aXNpb24sIEJFQSBTeXN0ZW1zIEluYzERMA8GA1UECxMIV0xDIEx1a2UxFTA
TBgNVB
AMTDGx1a2UuYmVhLmNvbTEfMB0GCSqGSIb3DQEJARYQZ2FyaW1lbHNAYmVhLmNvbTAeF
w0wMjA
xMDEwMDAwMDBaFw0wMzAxMDEwMDAwMDBaMIGOMQswCQYDVQQGEwJVUzETMBEGA1UECBM
KQ2Fsa
WZvcm5pYTEUMBIGA1UEChMLQkVBIFN5c3RlbXMxFTATBgNVBAsTDEVDSSBEaXZpc2lvb
jEfMB0
GA1UEAxMWbG9jYWxob3N0LnBlZXIxLWVuLmNydDEcMBoGCSqGSIb3DQEJARYNcGVlcjF
AYmVhL
mNvbTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC3+QMrol7OTZ7fWTacxAxqA+5ewLnp1
/XxhbQ
pGpBrz1Duj8fw7ybot04fkfQbJgRQEP6M3TyWBmoJAwMBkIbrAgMBAAGjGjAYMAkGA1U
dEwQCM
AAwCwYDVR0PBAQDAgXgMA0GCSqGSIb3DQEBBAUAA0EAA8QAs20bOFvebMd6mU6ui7lAY
Zd+5+d
OhTU0R03VgY35ZQXzyaOH7GtMHNOomFqKaRdckwAi75FZTuAfKVYJfw==
</X509Certificate>
  </X509Data>
</KeyInfo>

  </client-certificate>
</trading-partner>
```

Deleting Management Data

The Bulk Loader provides the ability to bulk delete management data. The delete operation removes trading partners information based on an input selector file. It deletes each selected leaf element and all linked child elements associated with that element. For example, if you delete a particular Trading Partner from the repository, all child certificate, binding, transport, and authentication elements are also deleted.

To delete management data using the Bulk Loader:

1. Create an input file that specifies the data elements to be deleted from the repository, as shown in the following example.

   TpmDelete.xml

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <trading-partner-management
       xmlns="http://www.bea.com/2003/03/wli/tpm"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.bea.com/2003/03/wli/tpm TPM.xsd">
       … elements to be deleted are specified here in XML …
   </trading-partner-management>
   ```

2. On a Windows system, open a command window.

3. In both Windows and UNIX, go to the following directory:

   ```
   BEA_HOME\wli_10.2\bin
   ```

   In the preceding line, *BEA_HOME* represents the WebLogic Platform home directory.

4. Execute the bulk delete by entering:

   ```
   bulkloader [-verbose] [-config <blconfig.xml>] [-wlibc] -delete
   <selector.xml>
   ```

   For a description of these options, see Table 5-1.

CHAPTER 6

Trading Partner Management Schema

This section describes the schema for Trading Partner Management (TPM) data that you can exchange with the TPM repository using:

- The WebLogic Integration Administration Console
- The Workshop TPM controls
- The Bulk Loader utility

TPM Overview

The TPM schema allows you to configure WebLogic Integration to share information among trading partners by defining the following:

- Addresses, phone and fax numbers
  - Address Element
- Authentications, encryptions, and certificates
  - Authentication Element
  - client-certificate Element
  - encryption-certificate Element
  - server-certificate Element
  - signature-certificate Element

- Protocol transports for RosettaNet, ebXML, and Web services
  - rosettanet-binding Element
  - rosettanet-service-defaults Element
  - web-service-binding Elementserver-certificate Element
  - ebxml-binding Element
- Data unique to your business needs
  - service Element
  - service-profile Element
  - signature-transforms Element
  - trading-partner Element
  - trading-partner-management Element
  - transport Element
  - xpath Element

A trading partner can have one or more service bindings that use different transport protocols for the exchange of documents. Each transport can use a variety of security authentication options, for client, server, signing, and messaging roles. The TPM schema allows you define the complete set of communication and configuration options for all trading partners.

Architecture: Trading Partners and Services

The root element of the TPM schema is the `trading-partner-management` element. The element provides logging and messaging options, and contains the two essential child elements for any configuration:

- `trading-partner`—a business entity that has authorization to send and receive business messages.

  The `trading-partner` element defines the settings for a single trading partner: authentication, security, and protocol options.

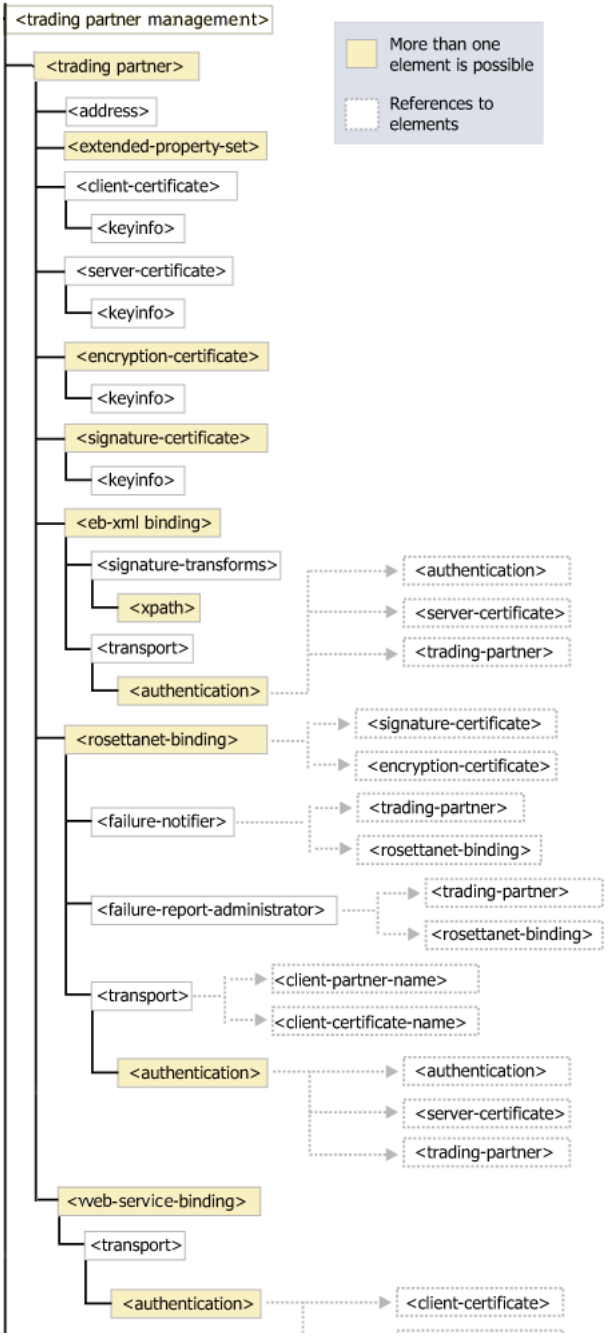
- `service`—a business process a trading partner offers

  The `service` element defines settings that describe how pairs of trading partners communicate: message protocols, message tracking, and RosettaNet service options.

The `service` element is rather simple and contains the following elements:

- `rosettanet-service-defaults`—for describing optional RosettaNet settings
- `service-profile`—for describing how pairs of trading partners communicate

The `trading-partner` element is far more complex. The following illustrations present the entity relationships among its elements.


<trading partner management>
More than one element is possible
References to elements
<trading partner>
<address>
<extended-property-set>
<client-certificate>
<keyinfo>
<server-certificate>
<keyinfo>
<encryption-certificate>
<keyinfo>
<signature-certificate>
<keyinfo>
<eb-xml binding>
<signature-transforms>
<xpath>
<transport>
<authentication>
<authentication>
<server-certificate>
<trading-partner>
<rosettanet-binding>
<signature-certificate>
<encryption-certificate>
<failure-notifier>
<trading-partner>
<rosettanet-binding>
<failure-report-administrator>
<trading-partner>
<rosettanet-binding>
<transport>
<client-partner-name>
<client-certificate-name>
<authentication>
<authentication>
<server-certificate>
<trading-partner>
<web-service-binding>
<transport>
<authentication>
<client-certificate>

Protocols and Security

The TPM schema provide configuration options for communication using the following service protocols:

- ebXML
- RosettaNet
- Web services available through JWS and JPD

The TPM schema provide settings for the authentication of trading partners as they send messages using these protocols at runtime for:

- Authentication credentials for outbound connections
- Mapping of trading partners to WebLogic Integration users for inbound connections
- Transport level security with a Secure Sockets Layer (SSL)
- Message level encryption and digital signatures

You configure these security and authentication options using:

- The `authentication` elements, that reside within a `transport` element for a given service protocol and allow clients and servers to authenticate.
- The individual service binding elements for each protocol, that provide settings for digital signatures and encryption for messaging.

The individual binding elements for each of the protocol services support non-repudiation by digitally signing outbound messages and acknowledgements based on the attributes that require signatures on messages and acknowledgement receipts. You can securely log message information as well.

The TPM schema supports the use of password aliases so you can refer to the password aliases in the WebLogic Integration password store. To learn more about password security, see Password Aliases and the Password Store section in *Using WebLogic Integration Administration Console* .

Extensibility

You can include custom information unique to your business needs using extended property sets. The extended-property-set allows any XML elements and attributes to be specified as child nodes

of the extended-property-set element. To learn more about extending TPM schema, see "extended-property-set Element" on page 6-22.

Test Mode

You can deploy your TPM options in a development environment without the need to specify explicit service profiles between trading partners. The test mode attribute on the `trading-partner-management` element allows you to test and deploy TPM business settings using the default bindings for your trading partners. This mode does not require separate service profiles to be set up for each pair of partners that exchange business messages.

To learn more about using test mode, see "trading-partner-management Element" on page 6-56.

Related Topics

Refer to Trading Partner Management section in *Using WebLogic Integration Administration Console* to learn more about using the WebLogic Integration Administration Console for TPM.

For more information about trading partner integration controls, see TPM Control, RosettaNet Control, and ebXML Control in *Using Integration Controls*.

For more information about using the Bulk Loader, see "Using the Trading Partner Bulk Loader" on page 5-1.

For more information aboutXML, see the W3C Recommendation, XML-Signature Syntax and Processing at the Web site of the W3C.

To learn more about the ebXML protocol, see the ebXML Collaboration-Protocol Profile and Agreement Specification - Version 2.0 at the Oasis Web site.

For more information about:

- ebXML in general, visit the ebXML Web site.
- RosettaNet protocol, visit the RosettaNet Web site.

Address Element

This element defines the external business address for a trading partner.

Syntax

```
<address>partnerMailAddress</address>
```

Attributes

None

Type

`xs:string`

References

To

None

Children

None

Hierarchy

Used By

trading-partner Element

Children

None

Authentication Element

This element specifies the authentication properties for a remote client that connects to the parent transport endpoint.

Syntax

```
<authentication>
       client-partner-name="tradingPartnerReference"
       client-authentication=      "BASIC
                                   |NONE
                                   |SSL_CERT_MUTUAL"
       username="loginName"
       password-alias="clientPassword"
       client-certificate-name="certificateReference"
       server-authentication=      "NONE
```

```
                                    |SSL_CERT"
    server-certificate-name="certificateReference"/>
```

Attributes

Table 6-1 Attributes

Attribute		
client-authentication	**Description**	Specifies whether to use client authentication, and if so, what kind.
	Allowable Values	BASIC—username and password NONE—no authentication SSL_CERT_MUTUAL—mutual SSL certificates
	Use	Optional
	Type	xs:NMTOKEN
	Default Value	None
client-certificate-name	**Description**	A reference to the name of the client certificate for mutual SSL authentication.
	Allowable Values	Any
	Use	Optional
	Type	reference
	Default Value	None

Table 6-1 Attributes

Attribute		
`client-partner-name`	**Description**	The name of the trading partner in the TPM repository to which the authentication applies.
	Allowable Values	Any
	Use	Required
	Type	`reference`
	Default Value	None
`password-alias`	**Description**	This is a reference to the password alias in the WebLogic Integration password store. The password is retrieved from the password store and is required when BASIC authentication is used.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None
`server-authentication`	**Description**	Specifies whether to use server authentication, and if so, what kind.
	Allowable Values	`NONE`—no authentication `SSL_CERT`—SSL certificate authentication
	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	No default value

Table 6-1 Attributes

Attribute		
`server-certificate-name`	**Description**	A reference to the name of the server certificate for SSL authentication.
	Allowable Values	Any
	Use	Optional
	Type	`reference`
	Default Value	None
`username`	**Description**	The user name for basic client authentication.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

References

To

client-certificate Element

server-certificate Element

trading-partner Element

From

None

Hierarchy

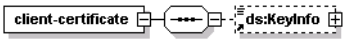
Used By

transport Element

Children

None

client-certificate Element



This element defines a digital certificate of a trading partner for client authentication access to a WebLogic Integration communication end point.

Syntax

```
<client-certificate
       name="certificateName"
       password-alias="keystoreEntryPasswordAlias">
       <ds:KeyInfo
       .
       .
       .
       </ds:KeyInfo>
</client-certificate>
```

Attributes

Table 6-2 Attributes

Attribute		
name	**Description**	The name for the client certificate in the TPM repository. The name is also the entry name in the local keystore.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None

Table 6-2 Attributes

Attribute		
password-alias	Description	This is a reference to the entry in the WebLogic Integration password store for the encrypted password. The encrypted password is used for accessing the password-protected keystore entry.
	Allowable Values	Any
	Use	Required
	Type	xs:string
	Default Value	None

References

To

None

From

Authentication Element

Hierarchy

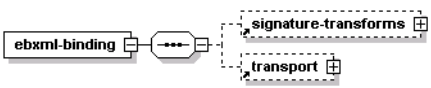
Used By

trading-partner Element

Children

ds:KeyInfo

ebxml-binding Element



This element defines the ebXML business protocol specific bindings of the parent trading partner.

The ebXML protocol supports non-repudiation by digitally signing outbound messages and acknowledgements based on the attributes `is-signature-required` and `is-receipt-signature-required`.

Syntax

```
<ebxml-binding
       business-protocol-name="protocolName"
       business-protocol-version="versionNo"
       delivery-semantics="[BESTEFFORT
                           |ONCEANDONLYONCE
                           |ATLEASTONCE
                           |ATMOSTONCE]"
       is-default="[true|false]"
       is-receipt-signature-require="[true|false]"
       is-signature-required="[true|false]"
       name="bindingName"
       persist-duration="intervalNo"
       retries="retriesNo"
       retry-interval="retryIntervalNo"
       signature-certificate-name="signatureCertificate">
       <signature-transforms
       .
       .
       .
       />
       <transport
       .
       .
       .
       />
</ebxml-binding>
```

Attributes

Table 6-3 Attributes

Attribute		
`name`	**Description**	The name for the binding in the TPM repository. A trading partner may have multiple `ebxml-binding` elements, so the name must be unique to the parent `trading-partner` element.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None
`business-protocol-name`	**Description**	Identifies the business protocol for message exchange.
	Allowable Values	`ebXML`
	Use	Optional
	Type	`xs:string`
	Default Value	None
`business-protocol-version`	**Description**	Identifies the version of the `business-protocol` name.
	Allowable Values	Any **Note:** Currently `1.0` and `2.0` are supported.
	Use	Optional
	Type	`xs:string`
	Default Value	None

Table 6-3 Attributes

Attribute		
`delivery-semantics`	**Description**	This attribute specifies reliable messaging behavior.
	Allowable Values	`BESTEFFORT`—best effort attempt to deliver messages. No reliable messaging. `ONCEANDONLYONCE`—Once and only once reliable messaging. Select this option for messaging that requires acknowledgement. `ATLEASTONCE`—at least once reliable messaging. Select this option for messaging that requires acknowledgement, but not duplicate elimination. `ATMOSTONCE`—at most once reliable messaging. Select this option for messaging that requires duplicate elimination, but not acknowledgement. For ebXML 1.0, only `BESTEFFORT` or `ONCEANDONLYONCE` are valid. For ebXML 2.0, all values are valid.
	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	False

Table 6-3 Attributes

Attribute		
`is-default`	**Description**	Identifies the default `ebxml-binding` for a trading partner in the event it has more than one.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	None
`is-receipt-signature-required`	**Description**	This setting, if true, specifies that the party who receives the ebXML messages from this trading partner through this binding must acknowledge them using the digitally signed receipt messages. The receipt messages must use the certificate of the acknowledging party. You can control the archival of signed receipts in a secure audit log by the global attribute `secure-audit-logging` in the root element `trading-partner-management`.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	None

Table 6-3 Attributes

Attribute		
`is-signature-required`	**Description**	This setting, if true, specifies that parties must digitally sign messages they send to the trading partner though this binding. You can control the archival of signed messages in a secure audit log by the global attribute `secure-audit-logging` in the root element `trading-partner-management`.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	None
`persist-duration`	**Description**	Specifies the duration for which messages have to be stored persistently for the purpose of duplicate elimination.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

Table 6-3 Attributes

Attribute		
`retries`	**Description**	Specifies the maximum number of times to attempt to send a reliably delivered message.
	Allowable Values	Any positive Integer
	Use	Optional
	Type	`xs:nonNegativeInteger`
	Default Value	`3`
`retry-interval`	**Description**	This attribute defines the time interval between attempts to send a reliably delivered message. The interval begins after the timeout period for message acknowledgement expires.
	Allowable Values	Time duration string
	Use	Optional
	Type	`xs:string`
	Default Value	None
`signature-certificate-name`	**Description**	References the name of the certificate for digitally signing messages.
	Allowable Values	Any
	Use	Optional This setting is required if the `is-signature-required` or `is-signature-receipt-required` attributes are true.
	Type	`reference`
	Default Value	None

Reference

To

signature-certificate Element

From

service-profile Element

Hierarchy

Used By

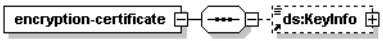
trading-partner Element

Children

signature-transforms Element

transport Element

encryption-certificate Element



This element defines a digital certificate for a trading partner for encrypting and decrypting exchanged messages.

Syntax

```
<encryption-certificate
       name="certificateName"
       password-alias="keystoreEntryPasswordAlias">
       <ds:KeyInfo
              .
              .
              .
       </ds:KeyInfo>
</encryption-certificate>
```

Attributes

Table 6-4 Attributes

Attribute		
`name`	**Description**	The name of the encryption certificate in the TPM repository. This name is also the entry name in the local keystore.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None
`password-alias`	**Description**	This is a reference to the entry in the WebLogic Integration password store for the encrypted password. The encrypted password is used for accessing the password-protected keystore entry.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

References

To

None

From

rosettanet-binding Element

Hierarchy

Used By

trading-partner Element

Children

ds:KeyInfo

extended-property-set Element

The `extended-property-set` element allows you to add custom XML nodes to your TPM configuration for your business needs.

The child elements are displayed within the repository as sub trees within an XML document, and can be nested.

```
<trading-partner name="ACMECORP" type="REMOTE" business-id="ACME-id">
       .
       .
       .
       <extended-property-set
              name="ACME Corp Extension"
              description="Contact Info"
              notes="the number format is important"/>
              <business-contact>Joe Smith</business-contact>
              <phone type="work">+1 123 456 7654</phone>
              <phone type="cell">+1 321 654 4567</phone>
              <city>Anytown</city>
              <state>California</state>
       </extended-property-set>
</trading-partner>
```

Syntax

```
<extended-property-set
       name="propertyName"
       description="propertyDescription"
       notes="propertyNotes">
       <xmlElement
       .
       .
       .
       </xmlElement>
```

```
</extended-property-set>
```

Attributes

Table 6-5 Attributes

Attribute		
`name`	**Description**	The name of the property set.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None
`description`	**Description**	A text description of the property set that appears in the WebLogic Integration Administration Console.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None
`notes`	**Description**	Text notes or documentation for the property set.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

References

To

None

From

None

Hierarchy

Used By

trading-partner Element

Children

Any

failure-notifier Element

This element represents the RosettaNet PIP failure notifier. It sends notification of failure (PIP0A1) messages to the appropriate trading partner and binding.

Syntax

```
<failure-notifier
       trading-partner-name="tradingPartnerReference"
       binding-name="bindingNameReference""/>
```

Attributes

Table 6-6 Attributes

Attribute		
`trading-partner-name`	**Description**	The name of the trading partner in the TPM repository that should receive RosettaNet failure notification.
	Allowable Values	Any
	Use	Required
	Type	`reference`
	Default Value	None

Table 6-6 Attributes

Attribute		
`binding-name`	**Description**	References the name of the service binding in the TPM repository for the provider.
	Allowable Values	Any
	Use	Optional
	Type	`reference`
	Default Value	None

References

To

rosettanet-binding Element

trading-partner Element

From

None

Hierarchy

Used By

rosettanet-binding Element

Children

None

failure-report-administrator Element

This element represents the RosettaNet PIP failure report administrator. It sends notification of failure (PIP0A1) messages to the appropriate trading partner and binding.

Syntax

```
<failure-report-administrator
       trading-partner-name="tradingPartnerReference"
       binding-name="bindingReference"/>
```

Attributes

Table 6-7 Attributes

Attribute		
trading-partner-name	**Description**	The name of the trading partner in the TPM repository that should receive RosettaNet failure notification.
	Allowable Values	Any
	Use	Required
	Type	reference
	Default Value	None
binding-name	**Description**	The name of the binding in the TPM repository for the provider.
	Allowable Values	Any
	Use	Optional
	Type	reference
	Default Value	None

References

To

rosettanet-binding Element

trading-partner Element

From

None

Hierarchy

Used By

rosettanet-binding Element

Children

None

reference simpleType

This references another element in the TPM repository.

Syntax

```
<reference>referenceName</reference>
```

Attributes

None

Type

```
xs:string
```

Hierarchy

Used By

Authentication Element

ebxml-binding Element

failure-notifier Element

failure-report-administrator Element

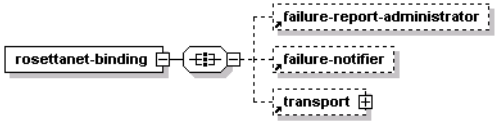
rosettanet-binding Element

service-profile Element

Children

None

rosettanet-binding Element



This element defines the RosettaNet business protocol specific bindings for the parent trading partner.

The RosettaNet protocol supports non-repudiation by digitally signing outbound messages and acknowledgements based on the `is-signature-required` and `is-receipt-signature-required` attributes.

Syntax

```
<rosettanet-binding
       name="bindingName"
       business-protocol-name="businessProtocolName"
       business-protocol-version="businessProtocolVersion"
       is-default="[true|false]"
       encryption-certificate-name="encryptionCertificateName"
       cipher-algorithm="[NONE|RC5|DES|TRIPLE_DES|RC2]"
       encryption-level="[NONE|PAYLOAD|ENTIRE_PAYLOAD]"
       is-signature-required="[true|false]"
       is-receipt-signature-required="[true|false]"
       signature-digest-algorithm="[SHA-1|MD5|None]"
       signature-certificate-name="signatureCertificateName"
       retries="noOfRetries"
       retry-interval="retryIntervalNo"
       process-timeout="processTimeoutNo">
       <failure-report-administrator/>
       <failure-notifier
       .
       .
       .
       />
```

```
<transport
.
.
.
/>
</rosettanet-binding>
```

Attributes

Table 6-8 Attributes

Attribute		
name	**Description**	The name for the binding in the TPM repository. A trading partner may have multiple `rosettanet-binding` elements, so the name must be unique to the parent `trading-partner` element.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None
business-protocol-name	**Description**	Identifies the business protocol for message exchange.
	Allowable Values	`RosettaNet`
	Use	Optional
	Type	`xs:string`
	Default Value	None

Table 6-8 Attributes

Attribute		
business-protocol-version	**Description**	Identifies the version of the `business-protocol` name.
	Allowable Values	`1.1` `2.0`
	Use	Optional
	Type	`xs:string`
	Default Value	None
is-default	**Description**	Identifies the default `rosettanet-binding` for a trading partner in the event it has more than one.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False
encryption-certificate-name	**Description**	The name of the encryption certificate for the encryption and decryption of messages.
	Allowable Values	Any
	Use	Optional
	Type	`reference`
	Default Value	None

Table 6-8 Attributes

Attribute		
`cipher-algorithm`	**Description**	The cipher algorithm for encrypting messages.

	Allowable Values	`NONE`
		`RC5`
		`DES`
		`TRIPLE_DES`
		RC2

	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	None
`encryption-level`	**Description**	This attribute determines how much of a message to encrypt.

	Allowable Values	`NONE`
		`PAYLOAD`
		`ENTIRE_PAYLOAD`

	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	None

Table 6-8 Attributes

Attribute		
`is-signature-required`	**Description**	This setting, if true, specifies that parties must digitally sign messages they send to the trading partner though this binding. You can control the archival of signed messages in a secure audit log by the global attribute `secure-audit-logging` in the root element `trading-partner-management`
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False

Table 6-8 Attributes

Attribute		
is-receipt-signature -required	**Description**	This setting, if true, specifies that the party who receives the RosettaNet messages from this trading partner through this binding must acknowledge them using the digitally receipt messages. The receipt messages must use the certificate of acknowledging party. You can control the archival of signed receipts in a secure audit log by the global attribute `secure-audit-logging` in the root element `trading-partner-management`.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False

Table 6-8 Attributes

Attribute		
`signature-digest-algorithm`	**Description**	This setting specifies the message digest algorithm used for the digital signature.

	Allowable Values	`SHA-1`
		`MD5`
		`None`
		If the value is `SHA-1`, `None`, or null, the Secure Hash Algorithm 1 (SHA-1), which produces a 160-bit hash, is used.
		If the value is `MD5`, the Message Digest 5 (MD5) message hash algorithm, which produces a 128-bit hash, is used.

	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	None
`signature-certificate-name`	**Description**	References the name of the certificate for digitally signing messages.
	Allowable Values	Any
	Use	Optional This setting is required if the `is-signature-required` or `is-signature-receipt-required` attributes are true.
	Type	`reference`
	Default Value	None

Table 6-8 Attributes

Attribute		
`retries`	**Description**	Specifies the maximum number of times to attempt to send a reliably delivered message.
	Allowable Values	Any positive Integer
	Use	Optional
	Type	`xs:nonNegativeInteger`
	Default Value	3
`retry-interval`	**Description**	This attribute defines the time interval between attempts to send a reliably delivered message. The interval begins after the time-out period for message acknowledgement expires.
	Allowable Values	Time duration string
	Use	Optional
	Type	`xs:string`
	Default Value	None
`process-timeout`	**Description**	The amount of time a PIP can be active before timing out.
	Allowable Values	Time duration string
	Use	Optional
	Type	`xs:string`
	Default Value	None

References

To

encryption-certificate Element

signature-certificate Element

From

failure-notifier Element

failure-report-administrator Element

service-profile Element

Hierarchy

Used By

trading-partner Element

Children

failure-report-administrator Element

failure-notifier Element

transport Element

rosettanet-service-defaults Element

This element specifies RosettaNet protocol-specific configuration attributes for a service.

Syntax

```
<rosettanet-service-defaults
       service-content-schema="schemaFilePath"
       use-dtd-validation="[true|false]"
       validate-service-content="[true|false]"
       validate-service-header="[true|false]"/>
```

Attributes

Table 6-9 Attributes

Attribute		
`service-content-schema`	**Description**	The XML schema for content validation. The service uses this schema only if `use-dtd-validation` is false and `validate-service-content` is true.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None
`use-dtd-validation`	**Description**	Specifies the kind of XML validation to perform. If true, the validation is from a DTD; if false, from XML schema.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False

Table 6-9 Attributes

Attribute		
`validate-service-content`	**Description**	Determines whether to validate the service content of all messages.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False
`validate-service-header`	**Description**	Determines whether to validate the service header for all messages.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False

References

To

None

From

None

Hierarchy

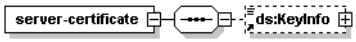
Used By

service Element

Children

None

server-certificate Element



This element defines a digital certificate for a trading partner to authenticate the identity of a target server for an outbound connection.

Syntax

```
<server-certificate
       name="serverCertificateName"
       password-alias="password-alias_1">
       <KeyInfo
       .
       .
       .
       </KeyInfo>
       </server-certificate>
```

Attributes

Table 6-10 Attributes

Attribute		
name	**Description**	The name of the server certificate in the TPM repository. The name is also the entry name in the local keystore.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None

Table 6-10 Attributes

Attribute		
`password-alias`	**Description**	This is a reference to the entry in the WebLogic Integration password store for the encrypted password. The encrypted password is used for accessing the password-protected keystore entry.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

References

To

None

From

Authentication Element

Hierarchy

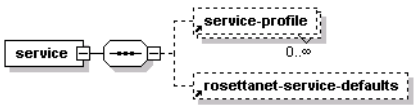
Used By

trading-partner Element

Children

`ds:KeyInfo`

service Element



This element represents a business process that a trading partner offers.

Syntax

```
<service
       name="serviceName"
       description="serviceDescription"
       notes="serviceNotes"
       service-type="[WEBSERVICE|PROCESS|SERVICECONTROL]"
       business-protocol="[WEBSERVICE|EBXML|ROSETTANET]">
       <service-profile
       .
       .
       .
       />
       <rosettanet-service-defaults
       .
       .
       .
       />
</service>
```

Attributes

Table 6-11 Attributes

Attribute		
name	**Description**	The name of the service in the TPM repository. The name corresponds to the name of a component on the local domain.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None

Table 6-11 Attributes

Attribute		
`description`	**Description**	A text description of the service that appears in the WebLogic Integration Administration Console.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None
`notes`	**Description**	Text documentation of the service element.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None
`service-type`	**Description**	The kind of service the element represents
	Allowable Values	WEBSERVICE—a JWS file PROCESSS—a JPD file SERVICECONTROL—a service control (JCX file)
	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	None

Table 6-11 Attributes

Attribute		
`business-protocol`	**Description**	The business protocol for the service, which determines the child service profile bindings.

	Allowable Values	`WEBSERVICE`
		`EBXML`
		`ROSETTANET`

	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	None

References

To

None

From

None

Hierarchy

Used By

trading-partner-management Element

Children

rosettanet-service-defaults Element

service-profile Element

service-profile Element

This element defines the interactions that two B2B trading partners agree to carry out, along with a specification for the business protocol implementation details such as messaging characteristics, security constraints, transport mechanisms, and workflow processes. Links to appropriate bindings for each trading partner specify these characteristics.

Syntax

```
<service-profile
       local-trading-partner="localTradingPartner"
       local-binding="localBinding"
       external-trading-partner="externalTradingPartner"
       external-binding="externalBinding"
       status="[ENABLED|DISABLED]"
       message-tracking="[NONE|DEFAULT|METADATA|ALL]"/>
```

Attributes

Table 6-12 Attributes

Attribute		
local-trading-partner	**Description**	This attributes references either: • the name of a local trading partner that hosts a JWS or JPD • the name of a local trading partner that uses a control to send messages to an external partner If you do not provide a value in the repository for this attribute, at runtime the value for this property comes from the `is-default` attribute.
	Allowable Values	Any
	Use	Optional
	Type	`reference`
	Default Value	None

Table 6-12 Attributes

Attribute		
`local-binding`	**Description**	References the name of the binding for the corresponding local trading partner. If you do not provide a value for this attribute, at runtime the value property comes from the binding with the `is-default` value of true.
	Allowable Values	Any
	Use	Optional
	Type	`reference`
	Default Value	None
`external-trading-partner`	**Description**	References the name of the trading partner with which the local trading partner interacts. This attribute can describe: • Remote trading partners • Collocated local trading partners
	Allowable Values	None
	Use	Required
	Type	`reference`
	Default Value	None

Table 6-12 Attributes

Attribute		
`external-binding`	**Description**	References the binding name for the corresponding `external-external-trading` partner.
	Allowable Values	Any
	Use	Optional
	Type	`reference`
	Default Value	None
`status`	**Description**	The deployed state of the service profile.
	Allowable Values	`ENABLED` `DISABLED`
	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	DIASABLED
`message-tracking`	**Description**	Determines whether to track messages, and if so, at what level.
	Allowable Values	`NONE`—no message tracking `DEFAULT`—default message tracking options `METADATA`—track message metadata `ALL`—track all message data
	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	Default

References

To

ebxml-binding Element

rosettanet-binding Element

trading-partner Element

web-service-binding Element

From

none

Hierarchy

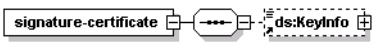
Used By

service Element

Children

None

signature-certificate Element



This element identifies a digital certificate for a trading partner and digitally signs messages for the associated trading partner.

Syntax

```
<signature-certificate
       name="signatureCertificateName"
       password-alias="certificatePasswordAlias">
       <KeyInfo
       .
       .
       .
       />
```

Attributes

Table 6-13 Attributes

Attribute		
`name`	**Description**	The name of the signature certificate in the TPM repository. This name is also the entry name in the local keystore.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None
`password-alias`	**Description**	This is a reference to the entry in the WebLogic Integration password store for the encrypted password. The encrypted password is used for accessing the password-protected keystore entry.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

References

To

None

From

ebxml-binding Element

rosettanet-binding Element

Hierarchy

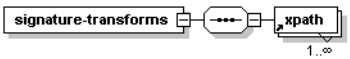
Used By

trading-partner Element

Children

`ds:KeyInfo`

signature-transforms Element



This element defines a sequence of optional XML data transformations for a digitally signed message, before WebLogic Integration signs the message. WebLogic Integration computes the message digest after performing transforms on the message.

Syntax

```
<signature-transforms>
       <xpath>xpath_expression-1</xpath>
       <xpath>xpath_expression-2</xpath>
       <xpath>xpath_expression-3</xpath>
</signature-transforms>
```

Attributes

None

References

To

None

From

None

Hierarchy

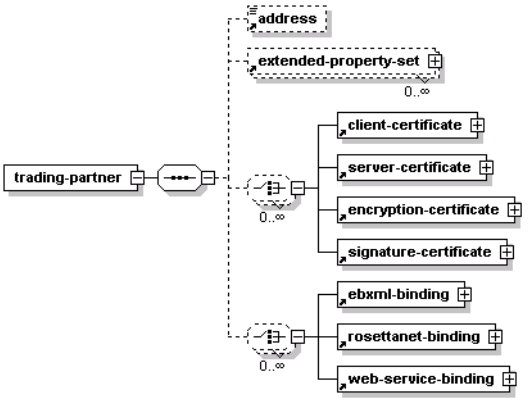
Used By

ebxml-binding Element

Children

xpath Element

trading-partner Element



A trading partner is a business entity with authorization to send and receive business messages in a conversation.

Syntax

```
<trading-partner
        name="tradingPartnerName"
        description="tradingPartnerDescription"
        notes="tradingPartnerNotes"
        status="[enabled|ENABLED|disabled|DISABLED]"
        type="[LOCAL|REMOTE]"
        is-default="[true|false]"
        business-id-type="businessIdType"
        business-id="businessId"
        email="emailAddress"
        phone="phoneNumber"
        fax="faxNumber"
        username="username">
        <address>partnerAddress</address>
        <extended-property-set>
        .
```

```
.
.
</extended-property-set>
<client-certificate>
.
.
.
</client-certificate>
<server-certificate>
.
.
.
</server-certificate>
<encryption-certificate>
.
.
.
</encryption-certificate>
<signature-certificate>
.
.
.
</signature-certificate>
<ebxml-binding>
.
.
.
</ebxml-binding>
<rosettanet-binding>
.
.
.
</rosettanet-binding>
<web-service-binding>
.
.
.
```

```
      </web-service-binding>
</trading-partner>
```

Attributes

Table 6-14 Attributes

Attribute		
name	**Description**	Name for the trading partner in the repository.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None
description	**Description**	A short text description of the trading partner that appears in the WebLogic Integration Administration Console.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None
notes	**Description**	Text notes or documentation of the trading partner.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

Table 6-14 Attributes

Attribute		
status	**Description**	A string that determines whether the trading partner is enabled to send and receive messages.

	Allowable Values	`enabled`
		`ENABLED`
		`disabled`
		`DISABLED`

	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	ENABLED
type	**Description**	Specifies whether the trading partner resides locally within WebLogic Integration domain or at an external remote location.
	Allowable Values	`LOCAL`—the trading partner resides within the domain `REMOTE`—the trading partner resides outside the domain
	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	Remote

Table 6-14 Attributes

Attribute		
is-default	**Description**	This setting indicates whether or not the trading partner is the default trading partner for sending and receiving messages for the local host system. This attribute can be set to true for trading partners with a `type` attribute of `LOCAL` only. Only one `LOCAL type` trading partner can have this value set to `true`.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False
business-id-type	**Description**	Identifies the type for naming convention for the associated `business-id` attribute. For example, a trading partner that is registered with Dun and Bradstreet might use a value of "`DUNS`".
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None
business-id	**Description**	Uniquely identifies the trading partner in message exchanges according to the `business-id-type`.
	Allowable Values	Any
	Use	Optional
	Type	`xs:string`
	Default Value	None

Table 6-14 Attributes

Attribute		
email	Description	An email address for the trading partner.
	Allowable Values	Any
	Use	Optional
	Type	xs:string
	Default Value	None
phone	Description	A telephone number for the trading partner.
	Allowable Values	Any
	Use	Optional
	Type	xs:string
	Default Value	None
fax	Description	A fax telephone number for a trading partner.
	Allowable Values	Any
	Use	Optional
	Type	xs:string
	Default Value	None
username	Description	The username in the WebLogic Integration security configuration that represents the trading partner.
	Allowable Values	Any
	Use	Optional
	Type	xs:string
	Default Value	None

References

To

None

From

Authentication Element

failure-notifier Element

failure-report-administrator Element

service-profile Element

Hierarchy

Used By

trading-partner-management Element

Children

Address Element

extended-property-set Element

client-certificate Element

server-certificate Element

encryption-certificate Element

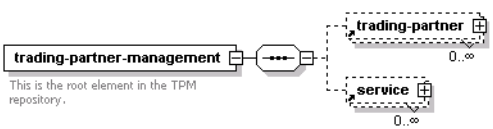
signature-certificate Element

ebxml-binding Element

rosettanet-binding Element

web-service-binding Element

trading-partner-management Element



This element is the document root for TPM. It serves as the parent element for all the major elements in the TPM repository.

Syntax

```
<trading-partner-management
       test-mode="[true|false]"
       message-tracking-default="[NONE|METADATA|ALL]"
       message-trace="[true|false]"
       message-trace-directory="directoryLocation"
       secure-audit-logging="[true|false]">
</trading-partner-management>
```

Attributes

Table 6-15 Attributes

Attribute		
message-tracking-default	**Description**	The default global setting for the message tracking level. The message tracking attribute of the `service-profile` element overrides this attribute.
	Allowable Values	`NONE`—no tracking `METADATA`—tracking message metadata `ALL`—all message data
	Use	Optional
	Type	`xs:NMTOKEN`
	Default Value	None

Table 6-15 Attributes

Attribute		
message-trace	**Description**	Toggles message tracing on and off.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	False
message-trace-directory	**Description**	The directory location where messages logs reside.
	Allowable Values	`false` `true`
	Use	Optional
	Type	`xs:boolean`
	Default Value	None
secure-audit-logging	**Description**	Specifies whether signed messages reside in a secured audit log.
	Allowable Values	True, false
	Use	Optional
	Type	`xs:boolean`
	Default Value	False

Table 6-15 Attributes

Attribute		
test-mode	Description	Specifies whether the repository is running in a test or production environment. In test-mode, you can send and receive messages between collocated trading partners without using service profiles.
	Allowable Values	false true
	Use	Optional
	Type	xs:boolean
	Default Value	True

References

To

None

From

None

Hierarchy

Used By

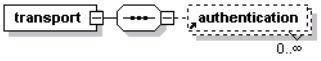
None

Children

trading-partner Element

service Element

transport Element



This element specifies the transport level properties and receiving endpoint for a binding.

Syntax

```
<transport
       protocol="[http|HTTP|https|HTTPS|jms|JMS]"
       protocol-version="[1.1|none]"
       endpoint="URL"
       timeout="timeoutNo">
       <authentication
       .
       .
       .
       />
</transport>
```

Attributes

Table 6-16 Attributes

Attribute		
`protocol`	**Description**	The protocol for sending and receiving messages. A value of `JMS/jms` is possible only when the transport is a child of the `web-service-binding` element.

	Allowable Values	`http`
		`HTTP`
		`https`
		`HTTPS`
		`jms`
		`JMS`

	Use	Required
	Type	`xs:NMTOKEN`
	Default Value	None

Table 6-16 Attributes

Attribute		
`protocol-version`	**Description**	The version of the transport protocol. This attribute is required for only HTTP/HTTPS protocols. The only supported version is 1.1.
	Allowable Values	"`1.1`" or no value
	Use	Optional
	Type	`xs:string`
	Default Value	None
`endpoint`	**Description**	The URL of the transport endpoint
	Allowable Values	Any
	Use	Optional
	Type	`xs:anyURI`
	Default Value	None
`timeout`	**Description**	The period that the transport waits until indicating that the transport of a message failed.
	Allowable Values	Time duration string
	Use	Optional
	Type	`xs:string`
	Default Value	None

References

To

None

From

None

Hierarchy

Used By

ebxml-binding Element

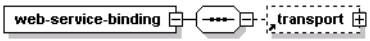
rosettanet-binding Element

web-service-binding Element

Children

Authentication Element

web-service-binding Element



This element and its child elements provide messaging properties such as transport endpoints, and authentication parameters for trading partners hosting or calling Web services.

Syntax

```
<web-service-binding>
       <transport
       .
       .
       .
       />
</web-service-binding>
```

Attributes

Table 6-17 Attributes

Attribute		
name	**Description**	The name for the binding in the TPM repository. A trading partner may have multiple `web-service-binding` elements, so the name must be unique to the parent `trading-partner` element.
	Allowable Values	Any
	Use	Required
	Type	`xs:string`
	Default Value	None

References

To

None

From

service-profile Element

Hierarchy

Used By

trading-partner Element

Children

transport Element

xpath Element

This element defines an Xpath expression that may be one of a sequence of optional XML data transformations on a message that it is to be digitally signed. The message digest is computed after any transforms are performed on the message.

Syntax

```
<xpath>xpath-expression</xpath>
```

Attributes

None

References

To

None

From

None

Hierarchy

Used By

signature-transforms Element

Children

None